IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Deboer et al.**	§	Group Art Unit: **2443**
	§	
Serial No. **10/718,297**	§	Examiner: **Sikri, Anish**
	§	
Filed: **November 20, 2003**	§	Confirmation No.: **9784**
	§	
For: **Extensible Mechanism for Executing Server Side Code**	§ §	

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY BRIEF (37 C.F.R. 41.41)

This Reply Brief is submitted in response to the Examiner's Answer mailed on April 28, 2010.

No fees are believed to be required to file a Reply Brief. If any fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447.

<u>RESPONSE TO EXAMINER'S ANSWER</u>

In response to the Examiner's Answer, Appellants note that the Examiner has cited new portions of *Christfort* in the Examiner's rejection of claims 1, 12, 29, and 32. Appellants address these new portions as follows:

With regard to the feature of <u>processing an input object identifying code for executing on one of a plurality of servers, said processing using a view list of at least one input object element, each input object element processing a type of code identified by the input object to output a deployable object</u>, the Examiner points to newly cited paragraphs [0059] and [0070]-[0072] of *Christfort* as teaching this feature. These cited sections teach that *Christfort* is directed to the creation and modification of on-line applications using an on-line software development kit, and that a middleware transformer is provided for transforming application output into output that is customized based on parameters or conditions associated with a service request. Thus, an application may produce several output variations, and the particular output displayed on the client device is based on parameters provided in the request.

The presently claimed invention recites an extensible mechanism for executing code on a server in three processing stages –view processing, a server processing, and a launcher processing. The view processing determines the code to run by processing an input object identifying code for executing on one of a plurality of servers, said processing using a view list of at least one input object element, each input object element processing a type of code identified by the input object to output a deployable object as recited in the claims. This outputted deployable object is then processed in the server processing. The server processing determines the server to execute the identified code by processing the deployable object using a server list of at least one server element to determine the one of the plurality of servers for executing the code, each server element enabling the deployable object to execute on a particular server and outputting a launchable object as recited in the claims. This outputted launchable object is then processed in the launcher processing. The launcher processing determines the client for interacting with the server to run the code by processing the launchable object using a launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers as recited in the claims. Thus, the presently claimed invention allows the use of different servers and launchers for executing different code types and provides a mechanism to easily add additional or modify existing servers, launchers, or code types.

Christfort makes no mention of providing a combination of a (1) view list of at least one input object element, a (2) server list of at least one server element to determine the one of the plurality of servers for executing the code, and a (3) launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers as claimed, nor of using the output object from a process using one list as input to another process using another list. With regard to the view list, *Christfort* simply teaches receiving parameters within a service request and providing a customized output based on the parameters. *Christfort* fails to teach that the customized output is provided to any other process that determines which server executes the code and the output from the other process is provided to another process that determines a client for launching the code on the determined server.

With regard to the feature of processing the deployable object using a server list of at least one server element to determine the one of the plurality of servers for executing the code, each server element enabling the deployable object to execute on a particular server and outputting a launchable object, the Examiner points to newly cited paragraphs [0076], [0090]-[0091], and [0093] of *Christfort* as teaching this feature. These cited sections of teach that an application provider may provide a link to another hosted or shared application that is associated with the host server. *Christfort* also teaches a shared hosted application may be created and saved at the application developer's website, and the user associates a URL with the shared hosted application. The shared hosted application is added as a "service" by providing a service name and the URL associated with the shared hosted application. *Christfort* further teaches that the application code for a hosted application and the code that causes the generation of a user interface may also be used to enter and edit the code are stored on one or more servers associated with the development provider. The user can save edited code in the interface which submits the code to a remote server, and the service name will be displayed in a list of existing services to users. End users may access the services via the host.

However, *Christfort* still does not teach processing the deployable object to determine the one of the plurality of servers for executing the code. No processing of a deployable object in order to determine the one of the plurality of servers for executing the code is disclosed as occurring in *Christfort*, as the target servers in *Christfort* are already known. For example, *Christfort* explicitly teaches that URL links are provided in the listed services to the particular associated application. No server determination is made *Christfort* to determine which server executes the code. Thus, while *Christfort* supports a plurality of servers for a hosting environment, *Christfort* does not support the explicitly recited feature of processing the deployable object to determine the one of the plurality of servers for executing the code as recited in the presently claimed invention. The *Christfort* process does not mention a processing step using a deployable object that determines which one of a plurality of servers that will execute the code. Instead, *Christfort* teaches a server side application environment where the user has already chosen the server to target prior to editing and saving the application. Thus, *Christfort* does not disclose a deployable object being processed to determine the one of the plurality of servers for executing the code. In addition, *Christfort* does not disclose a server element that enables the deployable object to execute on a particular server, or that the server element outputs launchable object. *Christfort* also makes no mention of an input object that outputs a deployable object for the type of code, and processing of the deployable object creates an output of a launchable object.

With regard to the feature of <u>processing the launchable object using a launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers</u>, the Examiner points to newly cited paragraphs [0094]-[0095] of *Christfort* as teaching this feature. This cited section teaches a process for accessing a new application or service. End users that can connect to the server network can access the service by manually selecting a service in a list of available services presented to the user.

However, a launchable object is defined in paragraph [0041] of the present specification as "an output, such as a Java code object, XML document etc., that contains information on how to access code to be run (i.e. a specific resource) on a particular server, and information about how to route access to the resource, (e.g. firewalls and/or the type of client application that can be used to access the web resource) if the access information is not sufficiently clear. For example, a URL may be constructed and provided to direct access to the resource when a Web browser is intended to consume the object." Thus, while *Christfort* supports any client launching the service application, *Christfort* makes no mention of pre-configuring a launcher at a client element to support launchable objects.

CONCLUSION

As shown above, the Examiner has failed to state valid rejections against any of the claims. Therefore, Appellants request that the Board of Patent Appeals and Interferences reverse the rejections. Additionally, Appellants request that the Board direct the Examiner to allow the claims.

/Cathrine K. Kinslow/

Cathrine K. Kinslow
Reg. No. 51,866
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Appellants